A. B. Veretennikov, Relevance ranking for proximity full-text search based on additional indexes with multi-component keys, *Vestn. Udmurtsk. Univ. Mat. Mekh. Komp. Nauki*, 2021, Volume 31, Issue 1, 132–148

DOI: https://doi.org/10.35634/vm210110

КОМПЬЮТЕРНЫЕ НАУКИ 2021. Т. 31. Вып. 1. С. 132–148.

УДК 519.683.5

© *А. Б. Веретенников*

РАНЖИРОВАНИЕ ДОКУМЕНТОВ ПРИ ПОЛНОТЕКСТОВОМ ПОИСКЕ С УЧЕТОМ РАССТОЯНИЯ С ИСПОЛЬЗОВАНИЕМ ИНДЕКСОВ С МНОГОКОМПОНЕНТНЫМИ КЛЮЧАМИ

Рассматривается задача полнотекстового поиска с учетом расстояния. Применение индексов с многокомпонентными ключами позволяет существенно ускорить обработку запросов, включающих часто встречающиеся слова, в сравнении с обычными инвертированными индексами. Было показано, что если запросы состоят из очень часто встречающихся слов, то время поиска может быть сокращено в 130 раз. В данной статье изучается влияние на точность поиска, выдачу в результатах поиска релевантных документов, архитектуры индексов с многокомпонентными ключами. Рассмотрен ряд методов определения релевантности документов разных авторов. Каждый метод применен при поиске в обычном индексе, а затем при поиске с использованием индексов многокомпонентных ключей. Результаты экспериментов подтверждают, что для ряда методов расчета релевантности поиск с использованием индексов многокомпонентных ключей предоставляет близкие результаты при сравнении с поиском в обычном индексе.

Ключевые слова: полнотекстовый поиск, поисковые системы, релевантность, инвертированные файлы, поиск с учетом близости слов, индексы с трехкомпонентными ключами.

DOI: 10.35634/vm210110

В системах полнотекстового поиска поисковым запросом является несколько слов. Результат поиска — это список записей, каждая из которых включает в себя информацию о документе, который содержит заданные слова, и о том, где в документе находятся эти слова. Список записей упорядочен. Наиболее релевантные записи, то есть, предположительно, наиболее точно соответствующие тому, что требуется пользователю, находятся в начале списка. При поиске с учетом расстояния в первую очередь важны документы, в которых слова поискового запроса располагаются вблизи друг от друга. Если слова запроса беспорядочно располагаются в произвольных местах документа, то такой документ менее отражает то, что нужно найти пользователю, чем документ, в котором слова запроса располагаются вблизи друг от друга, а значит, связаны друг с другом в контексте этого документа. При полнотекстовом поиске с учетом расстояния нужно хранить в индексах информацию о каждом вхождении каждого слова [1, 2]. Другими словами, используется word-level-индекс, а не document-level-индекс. Каждая запись в результатах поиска обычно содержит информацию о фрагменте текста документа, который включает в себя слова запроса и является таким фрагментом минимальной длины, а именно, позицию начала этого фрагмента в тексте и позицию его конца.

Слова в документах встречаются с разной частотой. Пример распределения частот встречаемости слов в текстах, иллюстрирующий закон Ципфа [3], изображен на рис. 1. Следовательно, скорость выполнения поискового запроса зависит от суммарного числа вхождений слов запроса в документах, и часто может быть так, что поисковая система обрабатывает запросы, состоящие из обычных слов, быстро, менее чем за секунду, а запрос, включающий часто встречающиеся слова, гораздо дольше, например за 20–30 с. Пример с использованием Lucene приведен в [4]. Пользователь может сказать, что система нестабильна. Поисковый запрос является видом простого запроса информации [5]. Если время

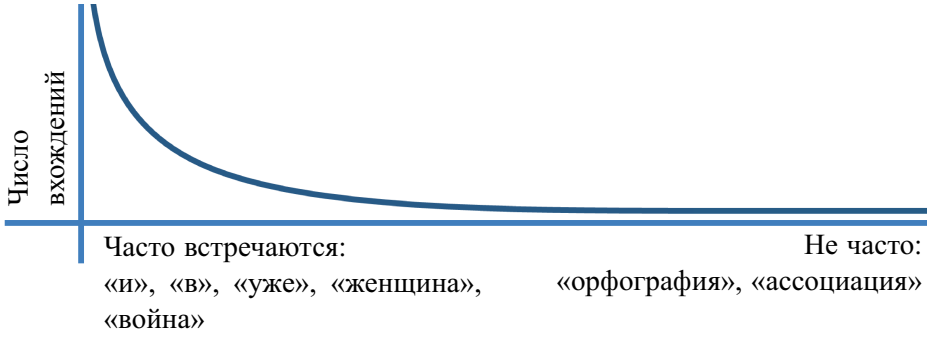

Рис. 1. Пример распределения частот слов в текстах

выполнения такого запроса больше одной секунды, непрерывность мышления пользователя может нарушаться, что снижает его производительность [5], а максимальное время выполнения запроса — две секунды. Можно наращивать вычислительные ресурсы. Другое решение — использование дополнительных индексов для выполнения проблемных запросов быстро. Автором работы предложен метод поиска с помощью дополнительных индексов [6–8]. В текущей работе показано, что при применении этих индексов при поиске имеем релевантные результаты. Третье решение — ограничение допустимых слов, которые можно искать. Вводится список стоп-слов, исключаемых из индекса и поиска. Этот подход ограничен, эти списки не могут быть большими. Четвертый подход — early-termination [9–12]. Данные в индексе упорядочены так, что при определенных методах расчета релевантности часть данных в индексе может быть определена как заведомо нерелевантная и пропущена при поиске. Но при поиске с учетом расстояния эти подходы не дают получить существенного результата [4].

Автором разработан метод поиска с помощью дополнительных индексов, с помощью которого скорость поиска была увеличена в 130 раз по сравнению с обычными инвертированными индексами, запросы состояли из очень часто встречающихся слов [4]. В [13] осуществлялся поиск произвольных запросов. Число словопозиций, считываемых при поиске с использованием дополнительных индексов, в среднем было меньше в 263 раза по сравнению с поиском в обычном инвертированном индексе. Значит, применение разработанных индексов перспективно, несмотря на то что время поиска может существенно зависеть от параметров индекса и текстовой коллекции.

Результаты текущей работы следующие. Показано, что, используя дополнительные индексы, мы получаем близкие результаты в смысле релевантности с результатами, полученными при поиске в обычном индексе. Эксперименты проведены с использованием общедоступной текстовой коллекции GOV2. Определен ряд методов расчета релевантности, при использовании которых результаты поиска, полученные при помощи дополнительных индексов, наиболее близки в смысле релевантности, с результатами, полученными при поиске в обычном индексе. Разработана методология для сравнения результатов поиска в дополнительных и в обычных индексах. Разработан метод определения статистических значений, требуемых для расчета релевантности, при поиске в дополнительных индексах. Определен метод двухэтапного поиска для текстовых коллекций, состоящих из малых документов.

Значительная часть работы посвящена теоретическим вопросам: обзор различных методов поиска и определение сильных сторон предлагаемых дополнительных индексов; выбор способа сравнения результатов поиска, полученных при использовании разных видов индекса; определение методов повышения качества поиска при использовании дополнительных индексов. Далее приводятся результаты экспериментов и их анализ.

Для полнотекстового поиска применяются инвертированные файлы [14, 15]. Для каждого вхождения каждого слова в документе в индексе сохраняется словопозиция — запись

вида (ID, P), где ID — идентификатор документа, P — позиция слова в документе. Инвертированный файл — это ассоциативный массив, позволяющий по заданному ключу, например слову, получать список словопозиций для данного ключа.

Ключом может быть пара слов, а значением — список словопозиций, соответствующих вхождению данной пары слов, непосредственно друг за другом, в текстах (next-word-индексы). Ключом может быть последовательность слов в тексте (фраза) [16]. Эти индексы ускоряют точный поиск фраз, когда ищут документы, содержащие заданную фразу, то есть все слова запроса в документе идут подряд, в соответствующем запросу порядке. Но, если в документе между словами запроса есть другие слова, этот документ не будет найден при поиске в таких индексах, поэтому их применение ограничено.

Можно ли исключать стоп-слова из поиска? Исключив из поиска часть часто встречающихся слов, проблемы с производительностью можно решить. Тем не менее современные поисковые системы, например Google, позволяют искать запросы, включающие все слова. Иногда часто встречающееся слово имеет особый смысл в контексте определенного запроса, и исключение его из поиска может приводить к непредсказуемым эффектам [4, 16]. Например, в запросе “who are you who”, последнее слово “who” — это название исполнителя песни “Who are You”. Также информация о стоп словах может применяться при расчете релевантности [17]. Цель нашего исследования — обеспечить быстрое выполнение любых запросов, поэтому включаем в индексы информацию обо всех словах текста.

При подходе early-termination [9–12] записи в определенном списке словопозиций упорядочены так, что в начале списка находятся записи релевантных документов, а в конце — нерелевантных. Это позволяет остановить чтение списка словопозиций на определенном этапе поиска, когда найдено достаточное число релевантных документов. Но для поиска с учетом расстояния это слабо применимо, так как всегда может найтись такой поисковый запрос и такой документ, слова запроса в котором располагаются вблизи друг от друга, но данные которого оказались в конце списка словопозиций.

Постановка задачи текущей работы следующая. В индексах с многокомпонентными ключами сохраняем для каждого слова информацию о других словах, располагающихся в тексте вблизи него, на расстоянии не более чем $MaxDistance$, где $MaxDistance$ — параметр, например, 5, 7, 9. Но что если в тексте слова запроса располагаются на большем расстоянии, например $MaxDistance = 9$, а искомые слова в тексте располагаются на расстоянии 10? Потенциальный результат будет пропущен при поиске. То есть при поиске с использованием дополнительных индексов можем получать меньше результатов, чем при поиске в обычном индексе. Важный вопрос — релевантны ли «пропускаемые» результаты. Возможно, результаты поиска, где слова запроса далеки друг от друга, не релевантны, не значимы и могут быть исключены. Тогда применимость индексов с многокомпонентными ключами будет обоснована.

В современных функциях определения релевантности это уже учтено. Например, если релевантность результата поиска для запроса из двух слов обратно пропорциональна квадрату расстояния между искомыми словами в тексте, то эта величина быстро убывает и приближается к нулю с ростом этого расстояния. Далее рассмотрим несколько методов расчета релевантности, предложенных разными авторами для поиска с учетом расстояния. Для каждого метода применим его в поиске с использованием обычного инвертированного индекса, затем в поиске с использованием наших индексов. Взяв результаты поиска в обычном индексе в качестве эталона, мы сравним с ними результаты поиска, полученные с помощью наших индексов, и определим, содержат ли последние все релевантные документы, полученные путем поиска в обычном индексе, и насколько полученные два набора результатов поиска близки в смысле релевантности.

§ 1. Методы определения релевантности документа

Результаты поиска сортируются, и документы в них располагаются от наиболее релевантных к менее релевантным. Это легко, если есть функция релевантности, ставящая в соответствие результату поиска число, например, BM25 [18], TF-IDF [19]. При учете расстояний между словами поискового запроса в тексте обычно считают, что релевантность документа обратно пропорциональная квадрату расстояния между словами запроса в тексте [20]. Для запроса из двух слов, где A, B — их позиции в тексте, $TP = 1/(A-B)^2$, где TP — релевантность результата в контексте близости слов запроса в тексте. Этот подход легко расширяется для запроса Q, состоящего из n слов [7]. Пусть $X = X_1, \ldots, X_n$ — позиции слов запроса в тексте. Тогда $TP(X) = 1 \Big/ \Big(|A(X) - B(X)| - (n-2)\Big)^2$, где $A(X) = \min\limits_{1 \leqslant i \leqslant n} X_i$, $B(X) = \max\limits_{1 \leqslant i \leqslant n} X_i$.

Релевантность документа можно определить следующим образом. Вначале отсортируем результаты поиска по значению TP. Затем каждый подсписок результатов поиска с одинаковым значением TP отсортируем в соответствии со значением BM25 или TF-IDF. Такие методы обозначим как $R_{TP,\text{BM25}}$ и $R_{TP,\text{TF-IDF}}$ соответственно.

С другой стороны, при поиске с учетом расстояния функцию релевантности можно определить как взвешенную сумму некоторых других функций. В [20] предлагается функция вида $R = \alpha \cdot SR + \beta \cdot IR + \gamma \cdot TP$, где $\alpha, \beta, \gamma \geqslant 0, \alpha + \beta + \gamma = 1$. Здесь SR — статический ранг документа, например PageRank [21]; IR учитывает слова запроса, например BM25 или TF-IDF; TP учитывает расстояние между словами запроса в тексте; α, β, γ — настраиваемые параметры, значения SR, IR, TP нормализованы, то есть находятся в диапазоне $[0, 1]$. Этот метод обозначим как $R_{WeiSum,\alpha,\beta,\gamma}$, или, с указанием значений параметров, $R_{WeiSum,0.1,0.4,0.5}$. При проведении экспериментов использовалась коллекция без рассчитанного SR, поэтому $\alpha = 0$. В качестве IR используем BM25.

Одни функции определены для документа в целом (например, BM25). Другие определены с учетом параметров фрагмента текста документа, содержащего слова запроса, например расстояния между словами запроса в тексте, их порядка. Тогда для одного документа возможны несколько результатов поиска с разными значениями релевантности. Если слова запроса встречаются в документе в двух разных местах, то имеем два результата с одинаковым BM25, но, может быть, разным TP. В списке результатов поиска возможны несколько записей для одного и того же документа, как с разным, так и одинаковым значением релевантности, в зависимости от способа расчета.

Можно определять релевантность с использованием интервалов. Один из методов — обработать все варианты вхождений слов запроса в тексте и рассчитать суммарный ранг [22], основан на подходах, представленных в [23,24]. Если $X = X_1, \ldots, X_n$ — позиции слов запроса в тексте, то $[A(X), B(X)]$ — интервал, включающий эти слова. Интервал должен быть минимальным, содержащим все искомые слова, но не содержащим в себе меньшего интервала, содержащего все искомые слова. В тексте может быть много удаленных друг от друга подходящих интервалов. Для каждого интервала подсчитывается свой ранг, эти значения суммируются для получения ранга документа. Финальный ранг рассчитывается для документа в целом. В общем случае рассматриваются также интервалы, содержащие часть искомых слов. Возможны разные подходы учета интервалов.

Выделим множества документов [23], $D_1, D_2, \ldots, D_{|Q|}$, где $|Q|$ — длина запроса; D_1 — множество документов, содержащих одно слово запроса, D_2 — документы, содержащие два слова запроса, и далее до $D_{|Q|}$ — документы, содержащие все слова запроса. Множества рассматриваются начиная с $D_{|Q|}$ и далее в обратном порядке. Если необходимое число документов найдено в $D_{|Q|}$, то остальные множества не нужны. То есть в первую очередь

важны документы, содержащие все слова запроса. Вес интервала $I = [p, q]$, для документа D, где p — начало I, q — конец: $Score_{\text{Clarke et al.}}(I, D) = \min(K/(q - p + 1), 1)$, где $K = 16$. Ранг документа определен суммой этих значений для всех найденных интервалов. Вес интервала здесь обратно пропорционален его длине (не квадрату длины).

Мы рассматриваем подобные функции расчета релевантности в ограниченном виде, учитывая только интервалы, содержащие все слова запроса. Учет всех возможных интервалов сложно реализовать с учетом того, что для обработки разных типов запросов (вводятся далее) применяются разные виды индексов и разные алгоритмы [4, 7]. Поскольку в [23] в первую очередь отбираются документы, содержащие все слова запроса, это ограничение не критично при анализе [23]. Обозначим этот метод как $R_{IntervalSum}$. Предполагаем, что если применимость индексов многокомпонентных ключей будет показана с этим ограничением, то и без него они будут показывать хороший результат.

В [24] алгоритм поиска интервалов основан на проходе текста слева направо. Слова запроса попадают в один интервал, если близки друг к другу в тексте, разные интервалы могут содержать в себе разное число слов. Для запроса Q и документа D

$$\text{BM25}(Q, D) = \sum_{e \in Q} W_e \frac{TF(D, e)(1 + k_1)}{TF(D, e) + K}, \text{ где } K = k_1 \times \left(1 - b + \frac{b \times |D|}{avg(|D|)}\right),$$

где k_1, b — параметры, $TF(D, e)$ — частота вхождения леммы или слова e в D (обычно $TF(D, e)$ — число вхождений e в D), $|D|$ — длина D (количество слов), $avg(|D|)$ — средняя длина документа в рассматриваемой коллекции документов. В качестве W_e можно брать IDF или другие варианты [22]. В [24] используется эта же формула, но $TF(D, e)$ заменяется на $rc(D, e)$, где rc определяется суммой рангов интервалов, включающих слово или лемму e. Ранг интервала определен $Score_{\text{Song et al.}}(I, D) = \frac{n_i^\lambda}{(q-p+1)^\gamma}$, похожим образом с [23], где n_i — число слов запроса, находящихся в интервале I, а λ, γ — параметры.

В [22] при расчете веса интервала учитываются IDF значения его крайних слов и также используются BM25 подобные конструкции. Применение BM25 подобных вариантов обосновывается ссылкой на [25], тем, что BM25 варианты хорошо работают. Вес интервала $I = [p, q]$ определен как $Score_{\text{Lu et al.}}(I, D) = W_l \cdot W_r \cdot (q-p+1)^{-2}$, где W_l, W_r — значения IDF для леммы, находящейся вначале интервала, и для леммы, находящейся в конце. В результате уменьшаются веса интервалов, на концах которых слова — часто встречающиеся. Ранг документа рассчитывается по BM25-подобной формуле, как в [24]. Пусть $\mathcal{I} = \mathcal{I}(Q')$ — набор интервалов, включающих в себя подзапрос Q'. Тогда

$$Score_{\text{Lu et al.}}(Q', D) = \frac{\sum_{I \in \mathcal{I}} Score_{\text{Lu et al.}}(I, D) \cdot (1 + k_1)}{\sum_{I \in \mathcal{I}} Score_{\text{Lu et al.}}(I, D) + K'}, \text{ где } K' = K \cdot \left(\sum_{e \in Q'} \min(W_e, 1)\right),$$

W_e — IDF элемента e. Пусть λ — параметр (используем 0.4, как в [22]). Вычисляя финальный ранг, для запроса Q рассматривают все его возможные подзапросы. Рассматриваются два множества, $\mathcal{Q}$ — множество всех подзапросов Q, $\mathcal{Q}'$ — множество подзапросов Q, порядок слов в которых соответствует порядку слов в Q.

$$Score_{\text{Lu et al.}}(Q, D) = (1 - \lambda) \cdot BM25(Q, D) + \lambda \cdot \sum_{Q' \in \mathcal{Q}} Score(Q', D) + \lambda \cdot \sum_{Q' \in \mathcal{Q}'} Score(Q', D).$$

Будем высчитывать приближенное значение данной функции, учитывая только интервалы, содержащие все слова запроса. Назовем данный метод $R_{IntervalOpt}$.

В современных системах часто применяется двухуровневый расчет релевантности [26, 27]. Последовательно применяются два механизма ранжирования. Первый формирует начальный набор документов, исключая заведомо нерелевантные документы. Могут использоваться уже рассмотренные функции, такие как BM25. Второй механизм обрабатывает начальный набор документов и вычисляет финальные ранги документов с учетом разных

признаков (features), таких как BM25, TF-IDF, длина документа, PageRank, позиция первого вхождения слова запроса в документе, расстояние между словами запроса в документе и так далее. Каждому документу или результату поиска соответствует вектор чисел. Второй механизм можно реализовать на основе машинного обучения, например искусственных нейронных сетей [28], генетических алгоритмов [29], градиентного бустинга [27]. При построении функции релевантности используется тестовая коллекция документов с рассчитанными векторами признаков и рангами документов для какого-то набора поисковых запросов. Построенная функция (например, нейронная сеть) позволяет вычислять финальный ранг уже произвольного документа, на основании его вектора признаков, для произвольного запроса. Первый механизм является быстрым с точки зрения производительности, но дает грубые результаты. Второй механизм более медленный, но позволяет определить релевантность с лучшим качеством.

В текущей работе изучаем функции релевантности, применимые для механизма первого уровня. Далее полученные оценки можно использовать при работе какого-либо механизма второго уровня. Механизм второго уровня базируется на данных механизма первого уровня. Если механизм первого уровня выдает близкие результаты при использовании дополнительных индексов с результатами, полученными при поиске в обычном индексе, то механизм второго уровня, если он есть, не требуется менять. Рассмотрим несколько функций расчета релевантности и для каждой осуществим эксперимент.

Выберем набор тестовых поисковых запросов. Для каждого запроса выполним его с использованием обычного инвертированного индекса, результаты отсортируем с учетом релевантности. Далее выполним запрос с использованием дополнительных индексов и результаты отсортируем с учетом релевантности. Сравним оба списка результатов. Если списки результатов похожи, то дополнительные индексы могут применяться с использованием рассматриваемого метода расчета релевантности и поиск будет осуществляться без потери качества. Рассматриваемые функции: $R_{TP,\text{BM25}}$, $R_{TP,\text{TF-IDF}}$, $R_{WeiSum,0,0.75,0.25}$, $R_{WeiSum,0,0.5,0.5}$, $R_{WeiSum,0,0.25,0.75}$, $R_{WeiSum,0,0.1,0.9}$, $R_{IntervalSum}$ — сумма рангов интервалов, $R_{IntervalOpt}$ — проверка BM25 подобных конструкций и сумма рангов интервалов. $R_{IntervalSumSq}$ — сумма рангов интервалов, как $R_{IntervalSum}$, но вес интервала определен как в $R_{WeiSum,0,0.75,0.25}$ (вес интервала обратно пропорционален не длине, а квадрату его длины). Эти функции соответствуют следующим подходам определения релевантности:

1) определяем список релевантных документов на основе того, насколько слова запроса близки друг к другу в тексте, далее уточняем релевантность с учетом функции, которая не зависит от позиций слов запроса в документе ($R_{TP,\text{BM25}}$ и $R_{TP,\text{TF-IDF}}$);

2) взвешенная сумма, в которой присутствует компонента, отвечающая за учет близости слов запроса в тексте, и компонента, значение которой не зависит от позиций слов запроса в тексте ($R_{WeiSum,0,0.75,0.25}$, $R_{WeiSum,0,0.5,0.5}$, $R_{WeiSum,0,0.25,0.75}$ и $R_{WeiSum,0,0.1,0.9}$);

3) значение релевантности определяется для всего документа на основании учета длин всех интервалов текста, включающих слова запроса ($R_{IntervalSum}$ и $R_{IntervalSumSq}$);

4) информация о длинах всех интервалов текста, включающих слова запроса, комбинируется с IDF информацией в стиле BM25 ($R_{IntervalOpt}$).

§ 2. Сравнение результатов поиска

Для запроса Q обозначим: $Ideal$ — список результатов поиска, полученный при поиске в обычном индексе; $Instance$ — список результатов поиска, полученный при поиске с использованием дополнительных индексов. Элемент списка результатов — это запись с полями: ID — идентификатор документа; P — позиция начала фрагмента текста, который содержит слова запроса; L — длина этого фрагмента текста, в словах; R — релевантность записи (число с плавающей точкой). Нужно сравнить два списка результатов как два вектора

и получить численное значение, которое будет говорить, насколько эти списки результатов различаются. Используем следующие методы сравнения: расстояние Левенштейна [30], $Precision$ [31], $NDCG$ (Normalized Discounted Cumulative Gain) [32, 33].

Для вычисления метрик нужно определить условия, при которых две записи X и Y вида (ID, P, L) равны: 1) $X.ID = Y.ID$; 2) $EP(X) = EP(Y)$.

$EP(V) = V.P$, если $V.L < LRD$, иначе $EP(V) = -1$, где LRD — параметр, используем $LRD = 50$. Смысл условия 2) в том, что если в тексте имеем искомые слова на очень большом расстоянии друг от друга (> 50), то конкретное значение этого расстояния не имеет существенного смысла. Поэтому если есть два результата поиска, относящиеся к одному файлу, и в обоих случаях слова в тексте очень далеки друг от друга (пусть даже и на разном расстоянии), то считаем эти два результата одинаковыми. Отметим, что современные поисковые системы для каждого результата поиска выдают фрагмент текста, в котором присутствуют слова запроса. Для Google длина такого фрагмента — примерно 15–30 слов. Данное определение важно, если применяем двухэтапный поиск (см. одноименный раздел далее) при использовании дополнительных индексов [6].

Рассмотрим число N и возьмем первые N элементов каждого списка, $Ideal_N$ и $Instance_N$. Сравним списки $Ideal_N$ и $Instance_N$. $Precision$ для конкретного N обозначим как $P@N$. При этом $Precision(N) = P@N = |Ideal_N \cap Instance_N|/|Instance_N|$.

Список $Ideal_N$, полученный при поиске в обычном индексе, считаем списком релевантных документов. И относительно него оцениваем релевантность списка $Instance_N$.

Для получения $P@N$ определяем число релевантных документов, входящих в список $Instance_N$, а именно количество таких элементов $Instance_N$, которые входят и в $Ideal_N$. Полученное значение делим на длину списка $Instance_N$. Расстояние Левенштейна также вычисляется между двумя списками, $Ideal_N$ и $Instance_N$, для конкретного N.

Значение DCG (Discounted Cumulative Gain) для конкретного N обозначим как $DCG@N$.

$$DCG@N = \sum_{i=1}^{N} \frac{2^{Rel(Instance[i])} - 1}{log_2(i+1)},$$ где $Rel(Instance[i])$ — численное значение релевантности конкретной записи списка $Instance$, который нумеруется с 1. Значение $Rel(x)$ определяется исходя из значений в списке $Ideal$, независимо от того, к какому списку результатов поиска принадлежит x. Пусть $x = (ID, P, L, R)$ — результат поиска, элемент списка $Instance$. Чтобы определить $Rel(x)$, ищем равную x запись по полям ID, P, L в $Ideal$ и берем значение релевантности R найденной записи.

Значение $IDCG$ (Ideal Discounted Cumulative Gain) для конкретного N обозначим как $IDCG@N$. Это значение вычисляется так же, как и $DCG@N$, но для $Ideal$.

$$IDCG@N = \sum_{i=1}^{N} \frac{2^{Rel(Ideal[i])} - 1}{log_2(i+1)},$$ где $Rel(Ideal[i])$ — численное значение релевантности i-й записи списка $Ideal$, который нумеруется с 1, то есть $Rel(Ideal[i]) = Ideal[i].R$.

$NDCG$ для конкретного N обозначим как $NDCG@N = (DCG@N)/(IDCG@N)$. Отметим, что $0 \leqslant NDCG@N \leqslant 1$. Если $NDCG@N$ близко к 1, значит, с помощью поиска с использованием дополнительных индексов мы получаем близкие по релевантности результаты по сравнению с поиском в обычном индексе.

Для $R_{TP,\text{BM25}}$ и $R_{TP,\text{TF-IDF}}$ значения релевантности (компонент R) в списках результатов поиска отсутствуют, потому что применяется двухуровневый процесс сортировки. Поэтому $Ideal[i].R$ мы переопределим как $1/i$ для этих методов.

$NDCG$ имеет преимущество перед первыми двумя метриками, при ее расчете учитываются численные значения релевантности документов. Представим, что в $Ideal$, первые два документа высокорелевантные, а остальные, скажем 100, имеют релевантность, близ-

кую к 0. Если, например, $N = 30$, то отсутствие низкорелевантных документов в $Instance$ не проблема. Такие случаи не учитываются при расчете $Precision$ и расстояния Левенштейна, и отсутствие большого числа низкорелевантных документов приведет к «ложному» ухудшению их значений, эти метрики будут показывать нам худшее качество поиска, чем на самом деле. А при расчете $NDCG$ эта проблема учитывается.

Если рассмотрим динамику изменения расстояния Левенштейна и $Precision$ в зависимости от N, то для малых N увидим, в какой степени при поиске с использованием дополнительных индексов мы получаем в результатах поиска те же высоко релевантные документы, что и при поиске в обычном индексе. То есть высокое значение $NDCG$ должно соответствовать высоким значениям расстояния Левенштейна и $Precision$ для малых N, при этом значения двух последних метрик при больших N могут снижаться, но это не влияет сильно на качество поиска.

Метрика $NDCG$ ключевая и часто используется в информационном поиске при сравнении списков результатов выполнения поисковых запросов с идеальными списками результатов. Последние часто формируются вручную, но в нашем случае мы берем в качестве идеального списка результатов те результаты поиска, которые получаем при поиске в обычном индексе. При проведении экспериментов выполняем некоторое количество поисковых запросов и вычисляем средние значения вышеуказанных метрик.

§ 3. Лемматизация и дополнительные индексы

Морфологический анализатор определяет для словоформы ее набор базовых форм или лемм. В словаре около 92 тысячи английских лемм. Пусть FL — список всех лемм, упорядочен по убыванию частоты встречаемости леммы в текстах. FL-номер леммы x, или $FL(x)$, — это ее номер в FL. Для лемм x, y определим $x < y$, если $FL(x) < FL(y)$. Назовем первые $SWCount$ элементов FL стоп-леммами, например “war”, “time”. Следующие $FUCount$ элементов FL — часто используемые леммы, например “beautiful”, “red”. Остальные леммы обычные, например “glorious”, “promising”. $SWCount$ и $FUCount$ — параметры. Эксперименты далее представлены для англоязычной коллекции GOV2 [34]. Используем только английский словарь, $SWCount = 500$, $FUCount = 1050$. Значение $SWCount = 500$ близко к 421, которое выбрано в [35]. Слово может иметь несколько лемм; например, слово “mine” имеет две леммы: $mine$ и my.

Применяем несколько видов дополнительных индексов [7]. Индекс трехкомпонентных ключей (f, s, t) — это список словопозиций леммы f, таких, что на расстоянии меньшем или равном $MaxDistance$ от f в тексте присутствуют леммы s и t. При этом f, s и t — стоп-леммы и $f \leqslant s \leqslant t$. Каждая словопозиция имеет вид $(ID, P, D1, D2)$, где ID — идентификатор документа, P — позиция леммы f в документе, например порядковый номер слова в документе, $D1$ — расстояние между f и s, $D2$ — расстояние между f и t.

Индекс двухкомпонентных ключей (w, v) — это список словопозиций леммы w, таких, что на расстоянии меньше или равном $MaxDistance$ от w в тексте присутствует лемма v. При этом w — часто используемая лемма, v — часто используемая или обычная лемма. Словопозиция имеет вид (ID, P, D), где ID — идентификатор документа, P — позиция леммы w в документе, D — расстояние между w и v.

Обычный индекс с NSW (near stop word) записями содержит в себе списки словопозиций для обычных и часто используемых лемм. Для каждой леммы x сохраняется список записей (ID, P, NSW), где ID — идентификатор документа, P — позиция x в документе, NSW — NSW запись, которая содержит в себе информацию обо всех стоп-леммах, которые в тексте присутствуют на расстоянии меньшем или равном $MaxDistance$ от P. Например, $(ID, P, NSW((war, 3), (time, -2)))$ означает, что на расстоянии (-2) от P в документе ID присутствует стоп-лемма $time$, а на расстоянии 3 от P присутствует war.

Ключевые моменты исследования следующие. В индексах сохраняем информацию о всех словах, включая часто встречающиеся. Применяем word-level-индексы. Применяем легко обновляемые индексы. Для одного ключа может быть несколько потоков данных. При поиске читаем весь список словопозиций для каждого требуемого ключа. Используем DAAT (Document-at-a-time) [17].

В связи с расчетом релевантности в порядок выполнения поискового запроса нужно внести изменения в отношении [7, 13]. При расчете релевантности нам нужны следующие значения. Для документа D и леммы x определим $TF(D,x)$ — число вхождений леммы x в документе. Для леммы x определим $DF(x)$ — число документов, содержащих x. Вычислять TF можем следующим образом. Если осуществляется чтение списка словопозиций x из обычного индекса с NSW-записями, то при его чтении мы можем подсчитывать $TF(D,x)$ для каждого документа, а также $DF(x)$ будет рассчитано после завершения чтения списка словопозиций. При этом NSW-записи хранятся в отдельном потоке данных и могут быть пропущены, если не требуются для конкретного запроса.

Для стоп-лемм и всех лемм x, с $FL(x) < TS$, используем таблицу DTA, в которой храним $TF(D,x)$ для каждого документа и каждой такой леммы x, а также $DF(x)$ для каждой такой леммы x. TS — это параметр, $TS \geqslant SWCount + FUCount$. Мы строим DTA при построении индекса и храним в оперативной памяти при поиске. Значение TS выбирается исходя из того, сколько оперативной памяти хотим выделить для таблицы.

Читая список словопозиций (f,s,t) или (w,v), нельзя определить ни $TF(D,x)$, ни $DF(x)$, так как (f,s,t) содержит только словопозиции f, где s и t были недалеко. Также и для стоп-лемм, словопозиции которых извлекаются из NSW-записей. В этих случаях используем данные из DTA. Также храним в оперативной памяти таблицу DL. Значение $DL(D)$ — длина документа D, то есть, суммарное число слов в документе D. Пусть $AvgDL$ — средняя длина документа и DC — количество документов. Для всех алгоритмов поиска $TF(D,x)$ и $DF(x)$ должны быть всегда доступны для всех лемм запроса для расчета BM25 и TF-IDF. Рассмотрим следующие виды запросов.

QT1. Все леммы запроса — стоп-леммы. Используем DTA для определения $TF(D,x)$ и $DF(x)$. Применяем индексы трехкомпонентных ключей (f,s,t).

QT2. Все леммы запроса — часто используемые леммы. Используем DTA для определения $TF(D,x)$ и $DF(x)$. Применяем индексы двухкомпонентных ключей (w,v).

QT3. Все леммы запроса — обычные леммы. Определяем $TF(D,x)$ и $DF(x)$ в процессе чтения списков словопозиций. Применяем обычный индекс с пропуском NSW-записей.

QT4. Запрос содержит какое-то количество стоп-лемм, а также какое-то количество часто используемых и/или обычных лемм. Выбираем часто используемую или обычную лемму w с наименьшей частотой встречаемости в текстах. Для этой леммы считываем информацию из обычного индекса с NSW-записями. Информация в NSW-записях позволяет учесть стоп-леммы, присутствующие в составе запроса.

Для любой не стоп-леммы v запроса, для которой индекс слова в запросе i не совпадает с индексом первого вхождения w в запросе, можем рассмотреть следующие варианты.

1. v — часто используемая. Для учета этой позиции вместо однокомпонентного индекса (v) используем (w,v) — список вхождений леммы v в текстах, таких, что v располагается вблизи w в тексте. Используем DTA для определения $TF(D,v)$ и $DF(v)$.

2. v — обычная, $FL(v) < TS$, а также в составе запроса есть часто используемая лемма x, находящаяся в запросе на индексе $j \neq i$. Используем (x,v)-индекс для учета леммы v. Используем DTA для определения $TF(D,v)$ и $DF(v)$.

3. v — обычная, $FL(v) \geqslant TS$. Используем обычный индекс (v), но пропускаем NSW-записи. Определяем $TF(D,v)$ и $DF(v)$ в процессе чтения списков словопозиций.

QT5. Запрос содержит хотя бы одну часто используемую лемму w и какое-то количество

обычных лемм. Случай QT5 похож на QT4, но в QT5 пропускаем все NSW-записи для всех лемм запроса, в остальном действуем аналогично. Альтернатива — метод основного элемента [13], который для QT5 заключается в следующем. Пусть w — часто используемая лемма запроса, с наименьшей частотой встречаемости. Для каждой другой леммы запроса v получим индекс (w, v), он содержит словопозиции леммы w такие, что в тексте недалеко была лемма v. Имея такой набор индексов, начинаем их считывать с начала. Данные в индексах упорядочены по возрастанию. Смещаемся в каждом из индексов таким образом, чтобы во всех из них текущая словопозиция имела одинаковое значение (ID, P). Если можем так сделать, значит, в тексте недалеко друг от друга находятся все леммы запроса. Формируем результат и продолжаем чтение индексов. В этом случае для v информация $TF(D, v)$ и $DF(v)$ определяема одним из следующих способов. Если $FL(v) < TS$, то используем данные из DTA. Если иначе, получим данные, считывая из обычного индекса с NSW-записями список словопозиций v. Чтение этого списка не влияет на результаты поиска, так мы только получаем данные $TF(D, v)$ и $DF(v)$. Это третий способ определения $TF(D, v)$ и $DF(v)$ для заданной леммы.

Рассмотрим запрос “Scalable Vector Graphics”. $FL(Scalable) = -1$, $FL(Vector) = 3227$, $FL(Graphics) = 1075$. Graphics — часто используемая лемма, остальные — обычные. Используем индексы (graphics, scalable), (graphics, vector) и индекс (scalable) для получения информации $TF(D, v)$ и $DF(v)$ для $v = scalable$ (NSW-записи пропускаем).

Таким образом, большее значение TS, например 5000, позволяет ускорить поиск.

§ 4. Двухэтапный поиск

Если искомые слова в тексте расположены на расстоянии большем $MaxDistance$, то дополнительные индексы не позволят найти такой текст. Сделаем два поиска, первый — с учетом расстояния, второй — без учета расстояния [6]. Первый поиск делаем, применяя дополнительные индексы, используются word-level-индексы. Во втором поиске применяем document-level-индекс, где для каждого слова документа сохранена информация только о первом вхождении. Подход применим, если документы большие, например порядка 300 Кб. Тогда поиск без учета расстояния работает быстро. BM25 [18], TF-IDF [19] вычисляются для документа в целом. Поэтому все релевантные документы в смысле этих функций будут точно найдены на втором этапе, а первый этап обеспечит нахождение всех документов, релевантных с точки зрения близости слов запроса в документе.

В обычном индексе с NSW-записями для каждого ключа храним список словопозиций (ID, P, NSW). Сохраним его в трех потоках данных, (ID), (P), (NSW), что позволит читать из индекса только нужную информацию. Если нужен поиск без учета расстояния, читаем поток (ID). Для леммы запроса v данные $TF(D, v)$ и $DF(v)$ сохраняем в памяти на первом этапе поиска, если $FL(v) \geqslant TS$, а если $FL(v) < TS$, то используется DTA. Для коротких списков словопозиций используем два потока данных (ID, P), (NSW), снижая число дисковых операций при создании индекса и поиске.

В обычном индексе с NSW-записями ключи — часто используемые и обычные леммы. Дополним этот индекс и стоп-леммами, но для стоп-лемм не сохраняем NSW-записи и для каждого документа сохраняем информацию только о первом вхождении. Так, двухэтапный поиск организуется для коллекций с достаточно большими документами.

Но в GOV2 документы небольшие, в среднем документ содержит 7 КБ текста, и поиск без учета расстояния может выполняться долго. Для часто встречающегося слова и большого документа при переходе от word-level-индекса к document-level-индексу поиск ускоряется существенно, так как вместо чтения длинного списка словопозиций мы читаем только одну. Чем меньше документ, тем меньше разница во времени поиска.

Для ускорения поиска применим еще оптимизацию. Для леммы v запроса Q выполняется $FL(v) < TS$. Выполним Q, исключив лемму v. Далее из результатов поиска нужно убрать все документы, не содержащие v. Документ не содержит лемму v, если $TF(D, v) = 0$. Таким образом, на втором этапе можно исключить из Q все леммы v, такие, что $FL(v) < TS$. В Q нужно оставить хотя бы одну лемму, то есть если для любой леммы запроса v выполнено $FL(v) < TS$, то мы исключаем из Q все леммы, кроме одной, той, которая реже всего встречается в документах (с максимальным FL-номером).

Рассмотрим запрос “Scalable Vector Graphics”. $FL(Scalable) = -1$, $FL(Vector) = 3227$, $FL(Graphics) = 1075$. Используем $TS = 5000$. Будет выполнен запрос “Scalable”, дополнительно документы, не содержащие слов “Vector” и “Graphics”, будут пропущены.

С использованием этой оптимизации скорость выполнения запроса на втором этапе существенно повышается, и двухэтапный поиск может быть применен и к коллекциям, состоящим из малых документов, таким как GOV2. Рассмотрим также дополнительную оптимизацию: word-level-поиск на втором этапе. Применяется, если число результатов поиска на первом этапе меньше определенного параметра, например 15. В этом случае усилим поиск на втором этапе. Будем исключать из состава запроса только стоп-леммы. Также поиск будем осуществлять не в document-level-индексе, а в word-level-индексе, то есть будем считывать все словопозиции обрабатываемых лемм запроса. По результатам экспериментов, время, затраченное на второй этап поиска, с использованием word-level-индекса, составило 14.8 % от общего времени поиска (оба этапа). В целом время, затраченное на второй этап поиска, составило 39 % от общего времени поиска.

§ 5. Результаты экспериментов

Используем текстовую коллекцию GOV2, 426 GB, 25 миллионов документов. После удаления HTML-тегов и разметки, имеем 167 GB чистого текста. Средняя длина текста документа 7 KB. Тестовый набор запросов включает в себя запросы из TREC Robust Task 2004 (250 запросов), TREC Terabyte Task с 2004 по 2006 (150 запросов), TREC Web Task с 2009 по 2014 (300 запросов) и TREC Terabyte Task 2006 Efficiency Topics (10000 запросов). Без дубликатов имеем всего 10690 запросов. Созданы два индекса:

1) обычный инвертированный индекс;

2) дополнительные индексы, включая индексы трехкомпонентных ключей (f, s, t), индексы двухкомпонентных ключей (w, v), NSW-записи, $MaxDistance = 12$.

В [36] изучались логи некой поисковой системы. Оказалось, в среднем пользователь просматривал 2.35 страниц результатов поиска, 58 % пользователей не просматривали более одной страницы. Большинство пользователей, 86 %, просматривали не более трех страниц. На одной странице показывалось 10 результатов. Число пользователей, просматривающих только одну страницу, увеличивалось со временем [37] и достигло 73 % для Web-поисковых систем из US. Также в [36] исследовалось, насколько длинные запросы применяют пользователи. Обнаружено, что средняя длина запроса равна 2.21 слова и запросы с длиной более чем 5 очень редки. Запросы длиной 6 составляют примерно 1 % всех запросов. Число всех запросов длиной более 6 — не более 4 % от общего числа запросов. Запросы длиной от 1-го до 3-х слов составляют 80 % от всех запросов.

Поэтому ограничим анализ первыми тремя страницами (30 первых результатов поиска) и внимательно изучим первую страницу результатов. В нашем наборе запросов запросов длиной до 3-х слов — 4710, запросов длиной до 5-и слов — 8788. В таблице 1 показаны средние значения расстояния Левенштейна, $Precision$ и $NDCG$ в зависимости от функций релевантности. Для расчета брались первые 10, а затем первые 30 результатов поиска. Первые три колонки со значениями в таблице соответствуют случаю, когда изучались первые

10 результатов поиска. Значение первой колонки рассчитано с учетом запросов длиной не более 3, второй — длиной не более 5, третьей — длиной не более 9. Следующие три колонки соответствуют случаю, когда изучались первые 30 результатов поиска. Аналогично случаю с 10-ю результатами поиска каждая колонка этой группы содержит значение, рассчитанное в зависимости от длины рассматриваемых запросов. Отдельно показаны данные для запросов длиной от 1-го до 3-х слов (с учетом [36]). Для этого узкого, но, с другой стороны, широкого случая, для всех рассматриваемых методов определения релевантности поиск в дополнительных индексах показывает хорошие результаты. Для $R_{WeiSum,0,0.1,0.9}$ имеем $NDCG@10 = 0.980$. Значения $NDCG@10$ для $R_{TP,\text{BM25}}$, $R_{TP,\text{TF-IDF}}$, $R_{WeiSum,0,0.5,0.5}$, $R_{WeiSum,0,0.25,0.75}$, $R_{WeiSum,0,0.1,0.9}$, $R_{IntervalSumSq}$ превышают 0.96. С ростом длины запроса значения метрик незначительно снижаются.

Таблица 1. Результаты экспериментов, $L = |Q|$

Функция	10, $L \leqslant 3$	10, $L \leqslant 5$	10, $L \leqslant 9$	30, $L \leqslant 3$	30, $L \leqslant 5$	30, $L \leqslant 9$
Среднее значение расстояния Левенштейна						
$R_{TP,\text{BM25}}$	0.542	1.308	1.621	2.264	4.701	5.474
$R_{TP,\text{TF-IDF}}$	0.497	1.264	1.571	2.096	4.527	5.283
$R_{WeiSum,0,0.75,0.25}$	1.298	1.863	1.964	5.042	6.615	6.765
$R_{WeiSum,0,0.5,0.5}$	1.039	1.585	1.722	4.162	5.773	6.039
$R_{WeiSum,0,0.25,0.75}$	0.847	1.373	1.535	3.588	5.190	5.550
$R_{WeiSum,0,0.1,0.9}$	0.751	1.262	1.449	3.293	4.955	5.410
$R_{IntervalSum}$	2.929	4.562	5.150	10.480	15.636	17.215
$R_{IntervalOpt}$	1.873	2.885	3.042	7.142	9.823	10.068
$R_{IntervalSumSq}$	1.352	2.325	2.579	5.817	8.576	8.989
Среднее значение $Precision$						
$R_{TP,\text{BM25}}$	0.968	0.909	0.887	0.958	0.900	0.886
$R_{TP,\text{TF-IDF}}$	0.971	0.913	0.892	0.960	0.904	0.890
$R_{WeiSum,0,0.75,0.25}$	0.917	0.875	0.870	0.903	0.867	0.867
$R_{WeiSum,0,0.5,0.5}$	0.938	0.898	0.891	0.925	0.890	0.886
$R_{WeiSum,0,0.25,0.75}$	0.954	0.917	0.908	0.941	0.908	0.902
$R_{WeiSum,0,0.1,0.9}$	0.962	0.929	0.918	0.951	0.918	0.912
$R_{IntervalSum}$	0.771	0.644	0.595	0.765	0.637	0.593
$R_{IntervalOpt}$	0.894	0.826	0.815	0.882	0.833	0.829
$R_{IntervalSumSq}$	0.919	0.845	0.826	0.895	0.828	0.818
Среднее значение $NDCG$						
$R_{TP,\text{BM25}}$	0.978	0.932	0.911	0.976	0.927	0.906
$R_{TP,\text{TF-IDF}}$	0.981	0.936	0.915	0.978	0.930	0.910
$R_{WeiSum,0,0.75,0.25}$	0.950	0.917	0.911	0.919	0.884	0.882
$R_{WeiSum,0,0.5,0.5}$	0.964	0.933	0.927	0.942	0.907	0.903
$R_{WeiSum,0,0.25,0.75}$	0.973	0.948	0.941	0.958	0.927	0.921
$R_{WeiSum,0,0.1,0.9}$	0.980	0.959	0.951	0.968	0.941	0.933
$R_{IntervalSum}$	0.826	0.730	0.684	0.800	0.696	0.653
$R_{IntervalOpt}$	0.948	0.906	0.896	0.909	0.860	0.853
$R_{IntervalSumSq}$	0.965	0.904	0.882	0.934	0.867	0.847

Значение $NDCG$ более близко к 1, чем $Precision$. Во-первых, $NDCG$ использовать более правильно, так как при расчете этого значения учитывается релевантность документов, то, что результаты в начале списка, важнее результатов, располагающихся в его конце.

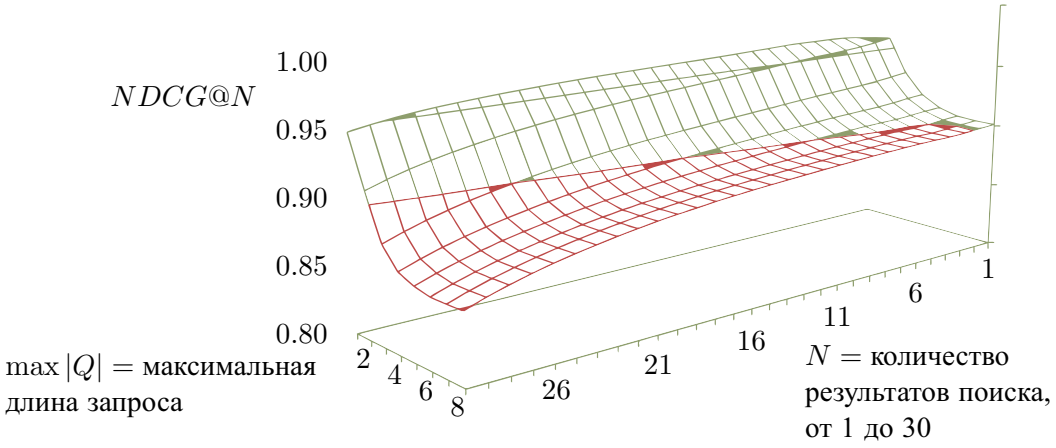

Рис. 2. Значения $NDCG@N$ для $R_{IntervalOpt}$

Во-вторых, если $NDCG$ больше, чем $Precision$, значит, при применении дополнительных индексов, если пропущен какой-то результат, в сравнении с поиском в обычном индексе, то, вероятно, этот результат не относится к самым релевантным результатам.

Лучшие результаты, вне зависимости от длины запроса, показывают $R_{TP,\text{BM25}}$, $R_{TP,\text{TF-IDF}}$, $R_{WeiSum,0,0.1,0.9}$, $R_{IntervalSumSq}$. Метод $R_{IntervalSum}$, в котором вес интервала обратно пропорционален длине интервала, работает существенно хуже методов, в которых вес интервала обратно пропорционален квадрату длины интервала, содержащего слова запроса. Предполагаем, что при использовании дополнительных индексов требуется применять методы, в которых используется квадратичная зависимость. В соответствии с [20] такой подход применяется в большинстве современных методов. Для метода взвешенной суммы $R_{WeiSum,0,x,y}$ значения расстояний Левенштейна, $Precision$ и $NDCG$, улучшаются при увеличении роста веса TP. При этом $NDCG@10$ для $R_{WeiSum,0,0.1,0.9}$ вне зависимости от длины запроса превышает 0.95. Это может быть связано с тем, что, когда мы считаем, что вес интервала обратно пропорционален квадрату длины интервала, значение TP уменьшается слишком резко. Возможно, следует, как и в [24], ввести вещественный параметр $\gamma \in [1, 2]$ и считать, что вес интервала обратно пропорционален его длине, возведенной в степень γ.

Для метода $R_{IntervalOpt}$ результаты значительно ухудшаются при рассмотрении более длинных запросов. Например, $NDCG@10$ для запросов длиной не более 3 равно 0.948, а если мы рассматриваем запросы длиной не более 9, то имеем уже 0.896.

Значения $NDCG@N$ показаны для $R_{IntervalOpt}$ на рис. 2 и для $R_{WeiSum,0,0.1,0.9}$ на рис. 3. На оси $\max|Q|$ показана максимальная длина запросов, рассмотренных при расчете $NDCG@N$. На оси N показано значение N — число результатов поиска, по которым рассчитано значение $NDCG@N$. Значение $NDCG@N$ показано в диапазоне от 0.8 до 1. $NDCG@N$ для $R_{IntervalOpt}$ снижается существенно при увеличении максимальной длины запроса, но от N оно зависит не сильно. Для $R_{WeiSum,0,0.1,0.9}$ значение $NDCG@N$ меньше зависит от максимальной длины запроса. Эти две функции сравниваем из следующих соображений. Самый худший результат показан при использовании $R_{IntervalSum}$, где вес интервала обратно пропорционален его длине. Этот подход мало применим для поиска с использованием дополнительных индексов. Следующий худший результат имеем для $R_{IntervalOpt}$, а для $R_{WeiSum,0,0.1,0.9}$ результаты лучше всего.

§ 6. Заключение

Разработана методология оценки качества поиска при поиске с использованием дополнительных индексов. Рассмотрев несколько подходов расчета релевантности, мы определили, что для ряда из них при поиске с использованием дополнительных индексов мы по-

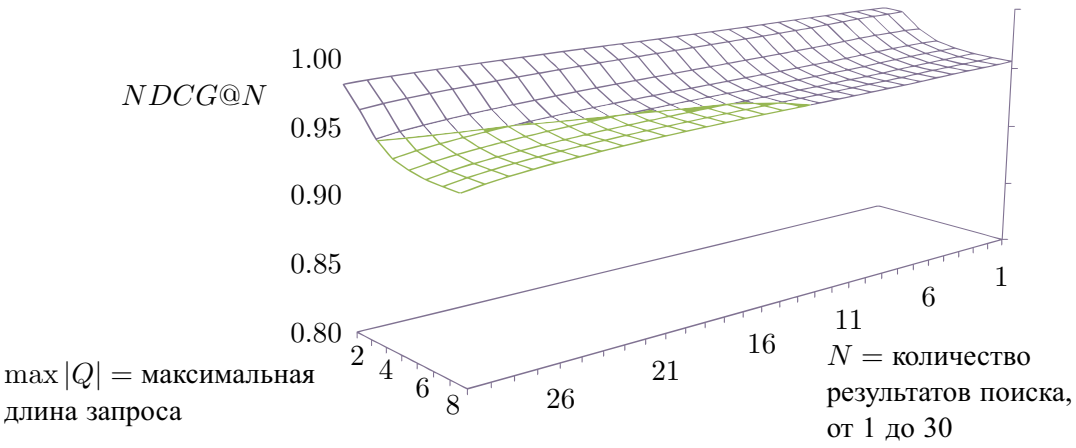

Рис. 3. Значения $NDCG@N$ для $R_{WeiSum,0,0.1,0.9}$

лучаем близкие в смысле релевантности результаты с результатами при поиске в обычном индексе. Наилучший результат показан при использовании функций $R_{TP,\text{BM25}}$, $R_{TP,\text{TF-IDF}}$, $R_{WeiSum,0,0.1,0.9}$, $R_{IntervalSumSq}$. Для $R_{WeiSum,0,0.1,0.9}$ при анализе первых 10-ти результатов поиска значение $NDCG$ не менее 0.95 при потенциальном максимуме 1. Метрика $NDCG$ является общепринятой метрикой для оценки качества поиска.

Для повышения качества поиска доработан метод двухэтапного поиска, который заключается в том, что мы вначале ищем с использованием дополнительных индексов, а далее ищем в обычном индексе, но либо без учета расстояния, либо учет расстояния осуществляется только для некоторых лемм запроса. Оба этих этапа в сумме занимают существенно меньше времени, чем поиск с учетом расстояния в обычном индексе.

В будущем интересны следующие задачи. Функция $R_{IntervalOpt}$ рассчитана с приближением, с учетом только интервалов, содержащих все слова запроса. Нужно разработать алгоритм, позволяющий при расчете релевантности учитывать и интервалы текста, содержащие часть слов запроса. Существенное время потребовала унификация расчета релевантности при наличии разных алгоритмов поиска, обрабатывающих разные виды запросов (QT1–QT5). Например, для QT1 есть отдельная группа алгоритмов [4]. Интересна задача разработки единого алгоритма поиска с использованием разработанных индексов. Интересен также анализ других функций определения релевантности.

Поступила в редакцию 11.10.2020

Веретенников Александр Борисович, к. ф.-м. н., доцент, кафедра вычислительной математики и компьютерных наук, Уральский федеральный университет, 620083, Россия, г. Екатеринбург, пр. Ленина, 51.
ORCID: https://orcid.org/0000-0002-3399-1889
E-mail: alexander@veretennikov.ru

Цитирование: А. Б. Веретенников. Ранжирование документов при полнотекстовом поиске с учетом расстояния с использованием индексов с многокомпонентными ключами // Вестник Удмуртского университета. Математика. Механика. Компьютерные науки. 2021. Т. 31. Вып. 1. С. 132–148.

COMPUTER SCIENCE 2021. Vol. 31. Issue 1. Pp. 132–148.

A. B. Veretennikov

Relevance ranking for proximity full-text search based on additional indexes with multi-component keys

Keywords: full-text search, search engines, relevance ranking, inverted indexes, proximity search, three-component key indexes.

MSC2020: 68P20, 68P10

DOI: 10.35634/vm210110

The problem of proximity full-text search is considered. If a search query contains high-frequently occurring words, then multi-component key indexes deliver improvement of the search speed in comparison with ordinary inverted indexes. It was shown that we can increase the search speed up to 130 times in cases when queries consist of high-frequently occurring words. In this paper, we are investigating how the multi-component key indexes architecture affects the quality of the search. We consider several well-known methods of relevance ranking; these methods are of different authors. Using these methods we perform the search in the ordinary inverted index and then in the index that is enhanced with multi-component key indexes. The results show that with multi-component key indexes we obtain search results that are very near in terms of relevance ranking to the search results that are obtained by means of ordinary inverted indexes.

REFERENCES

1. Sadakane K. Fast algorithms for k-word proximity search, *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, 2001, vol. 84, issue 9, pp. 2311–2318.
2. Gall M., Brost G. K-word proximity search on encrypted data, *30th International Conference on Advanced Information Networking and Applications Workshops (WAINA)*, 2016, pp. 365–372. https://doi.org/10.1109/WAINA.2016.104
3. Zipf G. K. *Selected studies of the principle of relative frequency in language*, Harvard University Press, 1932.
4. Veretennikov A. B. Proximity full-text search by means of additional indexes with multi-component keys: in pursuit of optimal performance, *Data Analytics and Management in Data Intensive Domains. DAMDID/RCDL 2018. Communications in Computer and Information Science*, vol. 1003, Cham: Springer, 2019, pp. 111–130. https://doi.org/10.1007/978-3-030-23584-0_7
5. Miller R. B. Response time in man-computer conversational transactions, *Proceedings of the December 9-11, 1968, fall joint computer conference, part I on - AFIPS '68 (Fall, part I)*, San Francisco, California, 1968, vol. 33, pp. 267–277. https://doi.org/10.1145/1476589.1476628
6. Veretennikov A. B. Proximity full-text search with response time guarantee by means of three component keys, *Bulletin of the South Ural State University. Series "Computational Mathematics and Software Engineering"*, 2018, vol. 7, issue 1, pp. 60–77 (in Russian). https://doi.org/10.14529/cmse180105
7. Veretennikov A. B. Proximity full-text search with a response time guarantee by means of additional indexes, *Intelligent Systems and Applications: Proceedings of the 2018 Intelligent Systems Conference*, Cham: Springer, 2018, pp. 936–954. https://doi.org/10.1007/978-3-030-01054-6_66
8. Veretennikov A. B. An efficient algorithm for three-component key index construction, *Vestnik Udmurtskogo Universiteta. Matematika. Mekhanika. Komp'yuternye Nauki*, 2019, vol. 29, issue 1, pp. 117–132 (in Russian). https://doi.org/10.20537/vm190111
9. Anh V. N., de Kretser O., Moffat A. Vector-space ranking with effective early termination, *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval - SIGIR '01*, 2001, pp. 35–42. https://doi.org/10.1145/383952.383957
10. Garcia S., Williams H. E., Cannane A. Access-ordered indexes, *ACSC '04 Proceedings of the 27th Australasian Conference on Computer Science*, Dunedin, New Zealand, 2004, pp. 7–14.

11. Lin J., Trotman A. Anytime ranking for impact-ordered indexes, *Proceedings of the 2015 International Conference on The Theory of Information Retrieval*, 2015, pp. 301–304. https://doi.org/10.1145/2808194.2809477
12. Anh V. N., Moffat A. Pruned query evaluation using pre-computed impacts, *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval - SIGIR '06*, 2006, pp. 372–379. https://doi.org/10.1145/1148170.1148235
13. Veretennikov A. B. Efficient full-text search by means of additional indexes of frequently used words, *Sistemy Upravleniya i Informatsionnye Tekhnologii*, 2016, vol. 66, issue 4, pp. 52–60 (in Russian).
14. Zobel J., Moffat A. Inverted files for text search engines, *ACM Computing Surveys*, 2006, vol. 38, issue 2, article 6. https://doi.org/10.1145/1132956.1132959
15. Yang Y., Ning H. Block linked list index structure for large data full text retrieval, *13th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD)*, 2017, pp. 2123–2128. https://doi.org/10.1109/FSKD.2017.8393099
16. Williams H. E., Zobel J., Bahle D. Fast phrase querying with combined indexes, *ACM Transactions on Information Systems (TOIS)*, 2004, vol. 22, issue 4, pp. 573–594. https://doi.org/10.1145/1028099.1028102
17. Crane M., Culpepper J., Lin J., Mackenzie J., Trotman A. A comparison of document-at-a-time and score-at-a-time query evaluation, *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, 2017, pp. 201–210. https://doi.org/10.1145/3018661.3018726
18. Robertson S. E., Walker S. Some simple effective approximations to the 2-Poisson model for probabilistic weighted retrieval, *SIGIR '94*, London: Springer, 1994, pp. 232–241. https://doi.org/10.1007/978-1-4471-2099-5_24
19. Salton G., Yang C. S. On the specification of term values in automatic indexing, *Journal of Documentation*, 1973, vol. 29, issue 4, pp. 351–372.
20. Yan H., Shi S., Zhang F., Suel T., Wen J.-R. Efficient term proximity search with term-pair indexes, *Proceedings of the 19th ACM international conference on Information and knowledge management - CIKM '10*, Toronto, 2010, pp. 1229–1238. https://doi.org/10.1145/1871437.1871593
21. Brin S., Page L. Reprint of: The anatomy of a large-scale hypertextual web search engine, *Computer Networks*, 2012, vol. 56, issue 18, pp. 3825–3833. https://doi.org/10.1016/j.comnet.2012.10.007
22. Lu X., Moffat A., Culpepper J. S. Efficient and effective higher order proximity modeling, *Proceedings of the 2016 ACM International Conference on the Theory of Information Retrieval*, 2016, pp. 21–30. https://doi.org/10.1145/2970398.2970404
23. Clarke C. L. A., Cormack G. V., Tudhope E. A. Relevance ranking for one to three term queries, *Information Processing and Management*, 2000, vol. 36, issue 2, pp. 291–311. https://doi.org/10.1016/S0306-4573(99)00017-5
24. Song R., Taylor M. J., Wen J.-R., Hon H.-W., Yu Y. Viewing term proximity from a different perspective, *Advances in Information Retrieval: Proceedings of 30th European Conference on IR Research*, Berlin: Springer, 2008, pp. 346–357. https://doi.org/10.1007/978-3-540-78646-7_32
25. Trotman A., Puurula A., Burgess B. Improvements to BM25 and language models examined, *Proceedings of the 2014 Australasian Document Computing Symposium on - ADCS '14*, 2014, pp. 58–65. https://doi.org/10.1145/2682862.2682863
26. Dang V., Bendersky M., Croft W. B. Two-stage learning to rank for information retrieval, *Advances in Information Retrieval: Proceedings of 35th European Conference on IR Research*, Berlin: Springer, 2013, pp. 423–434. https://doi.org/10.1007/978-3-642-36973-5_36
27. Silva S. D. N., de Moura E. S., Calado P. P., da Silva A. S. Effective lightweight learning-to-rank method using unified term impacts, *IEEE Access*, 2020, vol. 8, pp. 70420–70437. https://doi.org/10.1109/ACCESS.2020.2986943
28. Mitra B., Diaz F., Craswell N. Learning to match using local and distributed representations of text for web search, *Proceedings of the 26th International Conference on World Wide Web*, Republic and Canton of Geneva, Switzerland, 2017, pp. 1291–1299. https://doi.org/10.1145/3038912.3052579

29. Silva T. P. C., de Moura E. S., Cavalcanti J. M. B., da Silva A. S., de Carvalho M. G., Gonçalves M. A. An evolutionary approach for combining different sources of evidence in search engines, *Information Systems*, 2009, vol. 34, issue 2, pp. 276–289. https://doi.org/10.1016/j.is.2008.07.003
30. Levenshtein V. I. Binary codes capable of correcting deletions, insertions, and reversals, *Soviet Physics Doklady*, 1966, vol. 10, issue 8, pp. 707–710.
31. Makhoul J., Kubala F., Schwartz R., Weischedel R. Performance measures for information extraction, *Proceedings of DARPA Broadcast News Workshop*, 1999, pp. 249–252.
32. Järvelin K., Kekäläinen J. Cumulated gain-based evaluation of IR techniques, *ACM Transactions on Information Systems*, 2002, vol. 20, issue 4, pp. 422–446.
33. Liu T.-Y. *Learning to rank for information retrieval*, Berlin: Springer, 2011. https://doi.org/10.1007/978-3-642-14267-3
34. Büttcher S., Clarke C., Soboroff I. The TREC 2006 terabyte track, *Proceedings of the Fifteenth Text REtrieval Conference, TREC 2006*, 2006, pp. 128–141.
35. Fox C. A stop list for general text, *ACM SIGIR Forum*, 1989, vol. 24, issue 1–2, pp. 19–35. https://doi.org/10.1145/378881.378888
36. Jansen B. J., Spink A., Saracevic T. Real life, real users, and real needs: a study and analysis of user queries on the web, *Information Processing and Management*, 2000, vol. 36, issue 2, pp. 207–227. https://doi.org/10.1016/S0306-4573(99)00056-4
37. Jansen B. J., Spink A. How are we searching the World Wide Web? A comparison of nine search engine transaction logs, *Information Processing and Management*, 2006, vol. 42, pp. 248–263. https://doi.org/10.1016/j.ipm.2004.10.007

Received 11.10.2020

Veretennikov Aleksandr Borisovich, Candidate of Physics and Mathematics, Associate Professor, Department of Calculation Mathematics and Computer Science, Ural Federal University, pr. Lenina, 51, Yekaterinburg, 620083, Russia.
ORCID: https://orcid.org/0000-0002-3399-1889
E-mail: alexander@veretennikov.ru

Citation: A. B. Veretennikov. Relevance ranking for proximity full-text search based on additional indexes with multi-component keys, *Vestnik Udmurtskogo Universiteta. Matematika. Mekhanika. Komp'yuternye Nauki*, 2021, vol. 31, issue 1, pp. 132–148.